Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy [⋆]

Ali Khoshgozaran and Cyrus Shahabi

University of Southern California
Department of Computer Science
Information Laboratory (InfoLab)
Los Angeles, CA 90089-0781
[jafkhosh,shahabi]@usc.edu

Abstract. In this paper we propose a fundamental approach to perform the class of Nearest Neighbor (NN) queries, the core class of queries used in many of the location-based services, without revealing the origin of the query in order to preserve the privacy of this information. The idea behind our approach is to utilize one-way transformations to map the space of all static and dynamic objects to another space and resolve the query *blindly* in the transformed space. However, in order to become a viable approach, the transformation used should be able to resolve NN queries in the transformed space accurately and more importantly prevent malicious use of transformed data by untrusted entities. Traditional encryption based techniques incur expensive $O(n)$ computation cost (where n is the total number of points in space) and possibly logarithmic communication cost for resolving a KNN query. This is because such approaches treat points as vectors in space and do not exploit their spatial properties. In contrast, we use Hilbert curves as efficient one-way transformations and design algorithms to evaluate a KNN query in the Hilbert transformed space. Consequently, we reduce the complexity of computing a KNN query to $O(K \times \frac{2^{2N}}{n})$ and transferring the results to the client in $O(K)$, respectively, where N, the Hilbert curve degree, is a small constant. Our results show that we very closely approximate the result set generated from performing KNN queries in the original space while enforcing our new location privacy metrics termed *u-anonymity* and *a-anonymity*, which are stronger and more generalized privacy measures than the commonly used K-anonymity and cloaked region size measures.

1 Introduction

An important class of spatial queries consists of nearest-neighbor (NN) query and its variations. These queries search for data objects that minimize a distance-based function with reference to one or more query objects (e.g., points). In location-based services, a group of mobile users want to find the location of their K closest objects to their current

[⋆] This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0238560 (PECASE), IIS-0324955 (ITR), and unrestricted cash gifts from Google and Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

location (KNN). One obvious requirement with KNN queries is that the location of the query point(s) needs to be known in order to perform the query. However, in many applications such as in location-based services, a user may not want to reveal its location in order to preserve his/her privacy.

In this paper, we propose blind evaluation of Nearest Neighbor queries in order to preserve users' location from being revealed to location servers addressing such queries. For clarity reasons, for the rest of this paper, we will focus on location-based services in the 2-D space as the motivating application since it is clear that the query point is identical to the user location and hence its hiding preserves user's location privacy. While this application by itself is important enough to justify this research effort, we believe that the blind evaluation of KNN queries is fundamental and core to many other privacy preserving applications in sensor networks, online mapping services, geospatial information systems and numerous other applications in geospatial decision making.

Protecting users locations while responding to a KNN query is challenging due to the fact that there is an interesting dilemma in resolving such queries: while precise query location is needed to generate the result set for a KNN query, the privacy constraints of the problem does not allow revealing users' location information to the untrusted entity responding to such queries. In order to resolve this dilemma, we propose a fundamental approach based on utilizing the power of one-way transformations to preserve users' location privacy by encoding the space of all static and dynamic objects and answering the query blindly in the encoded space.

There is an inherent limitation in using traditional encryption techniques for blind evaluation of KNN queries. To illustrate, assume our server uses recently proposed encryption techniques to compute the encryption of the Euclidean distance between an encrypted point (i.e., the query origin) and each point of interest [8]. These encrypted distances can then be sent back to the client who can decrypt them and find the top K results. Trivially, this protocol satisfies our definition of blind KNN evaluation (see Section 3) since the location of neither the query point nor the result set is revealed to the server. However, the main limitation here is that distance between query point and each and every point of interest must both be computed and transferred to the client, i.e., $O(n)$ computation and communication complexity where n is the size of the database. There are cryptographic binary search communication protocols that may reduce the communication complexity to logarithmic; however, the computation complexity at the server cannot be reduced further. This is because the points of interest are treated as vectors with no exploitation of the fact that they are in fact points in space. Instead, we use Hilbert curves to transform original space to an encoded space stored at the server. Consequently, the server's encoded space still has the property that the nearby points stay close to each other and hence can reduce the KNNs computational complexity to $O(K \times \frac{2^{2N}}{n})$ where N, the curve order, is a small constant. Moreover, since only the K closest points are sent back to the client, the communication complexity becomes $O(K)$. We also introduce two new location privacy metrics termed user-based anonymity or *u-anonymity* and area-based anonymity or *a-anonymity*. These metrics are stronger and more generalized than the privacy measures commonly used by the K-anonymity and spatial cloaking based approaches. We analytically prove that our technique satisfies these two stronger privacy metrics.

We have performed several experiments to evaluate the effectiveness of our approach. As detailed in Section 7, we show that our proposed technique achieves a very close approximation of performing KNN queries in the original space by generating a result set whose elements on average have less than 0.08 mile displacement to the elements of the actual result set in a 26 mile by 26 mile area containing more than 10000 restaurants. We also show that a malicious attacker gains almost no useful knowledge about the parameters of our encoding techniques, even when significant knowledge about the key is compromised. In other words, a nominal displacement error in approximating only one of the key parameters, (a meter displacement in a 670 square mile area) results in no useful information for compromising our encryption scheme.

We stress that our technique does not always generate the exact ground-truth answer for a query because of its nature of reducing the dimensionality of data. However we believe there are many use case scenarios in location-based services where a *satisfactory* approximation of the result is still useful as long as users' privacy is preserved.

2 Related Work

The closest set of studies to our work is the class that preserves user location privacy using the *cloaking* techniques. With this approach, a trusted *anonymizer* is usually in charge of receiving user's precise location information and trying to disguise it by blurring user's exact location by (for example) extending it from a *point* location to an *area* (spatial extent) and sending a region containing several other users instead of a point to the server. A similar approach based on the concept of *K-anonymity* is extending the *cloaked* area until it is large enough to include a minimum of $K-1$ other users. Hence, the user's location cannot be distinguished from the location of the other $K-1$ users in the same extended area. This extended area will then be used to resolve spatial queries such as NN queries. Several techniques based on cloaking and K-anonymity have been proposed in the literature to reduce the resolution of the user's location information [1, 2, 4–6, 13, 14].

Cloaking and K-anonymity approaches have some limitations. First, by design cloaking relies on a trusted entity to "anonymize" users' locations which means all queries should trust the *anonymizer* during the system's normal mode of operation. The anonymizer will also become a single point of failure and a potential scalability bottleneck as several handshakes must occur between the user and anonymizer to exchange user profiles and anonymity measures. Another limitation of cloaking techniques in general is that either the quality of service or overall system performance degrades significantly as users choose to have more strict privacy preferences. For example, if the user requires a better K-anonymity, the system needs to increase K for that user which would result in a larger cloaked area and hence less accurate query response. Alternatively, if one requires to maintain the quality of service the location server has to resolve the spatial query for each and every point in the cloaked region and send the entire bulky result to the anonymizer to be filtered out. This will obviously affect the overall system performance, communication bandwidth and server throughput and results in more sophisticated query processing. Finally, the concept of K-anonymity does not work in all scenarios. For example, in a less populated area, the size of the extended area can be prohibitively large in order to include $K - 1$ other users. Some studies try to address this limitation by proposing more robust ways of determining the area of cloaking [11]. However, they will still need a trusted anonymizer to be able to respond to user queries and in the most optimistic scenario will reveal the region a user is located in, to an untrusted location server. In Section 6, we explain how our proposed approach eliminates the need for an anonymizer and why the accuracy of the result only depends on the quality of the transformation and remains consistent for all users.

3 Preliminaries

In this section, we first formally define the problem of blindly evaluating a KNN query and briefly discuss our approach and its use of one-way transformations. We also study the challenges associated with finding the right transformations and review an important class of many-to-one dimensional mappings called *space filling curves* which are used in our approach to achieve location privacy.

3.1 Formal Problem Definition

Given a set of static objects $S = (o_1, o_2, \ldots, o_n)$ in 2-D space, a set of users $U = (u_1, u_2, \ldots, u_M)$ and a set of dynamic query points $Q = (q_1, q_2, \ldots, q_m)$, the KNN query with respect to query point q_i finds a set $S' \subset S$ of K objects where for any object $o' \in S'$ and $o \in S - S'$, $D(o', q_i) \leq D(o, q_i)$ where D is the Euclidean distance function. In a typical KNN query scenario, the static objects represent points of interest (POI) and the query points represent user locations. We now define some of the properties a location server should posses in order to enable user location protection while responding to a KNN query.

Definition 1. u-anonymity: While resolving a KNN query, the user issuing the query should be indistinguishable among the entire set of users. In other words, for each query Q, $P(Q) = \frac{1}{M}$ where $P(Q)$ is the probability that query Q is issued by a user u_i and M is the total number of users. Note that this definition ensures the server does not know which user queried from a point q_i; however, we also need to ensure that the server does not know which point the query Q is issued from. This requirement is captured in Definition 2.

Definition 2. a-anonymity: While resolving a KNN query, the location of the query point should not be revealed. In other words, for each query Q, $P'(Q) = \frac{1}{area(A)}$, where A is the entire region covering all the objects in S, and $P'(Q)$ is the probability that query Q was issued by a user located at any point inside A.

Note that Definitions 1 and 2 impose much stronger privacy requirements than the commonly used K-anonymity [1, 4–6, 11, 13, 14, 18], in which a user is indistinguishable among K other users or his location is blurred in a cloaked region R. The above definitions of location privacy are free of metrics such as K and R. They are in fact identical to an extreme case of setting $R = A$ for spatial cloaking, and an extreme case of setting $K = M$ for K-anonymity.

Definition 3. Result set anonymity: The location of all points of interest in the result set should be kept secret from the location server. More precisely $\dot{P}(o_j) = 1/n$ for $j = 1 \ldots n$ where $\dot{P}(o_j)$ is the probability that o_j is a member of the result set for query Q and n is the total number of POI's.

Definition 4. Blind evaluation of KNN: We say a KNN query is blindly evaluated if the *u-anonymity, a-anonymity* and *result set anonymity* constraints defined above are all satisfied. In other words, in blind evaluation of KNN, the identity and location of the query point as well as the result set should not be revealed. We term our approach *blind evaluation of KNN queries* because it attempts to prevent any leak of information to essentially blind the server from acquiring information about a user's location. For the rest of the paper, we use the term *user* to refer to the user located at query point P issuing the query Q. The following example shows how the above properties should be satisfied in a typical KNN query. Suppose a user asks for his 3 closest gas-stations. In this case a *malicious entity* should acquire neither the location of the user (i.e., a-anonymity) not its identity (i.e., u-anonymity) nor the actual location or identity of any of the 3 closest gas stations in the response set (i.e., result set anonymity) while the user should receive the actual points of interest matching his query.

Based on the above properties, we term a location server *privacy aware* if it is capable of blindly evaluating a KNN query while providing accurate results. The challenge in blind evaluation of KNN queries is that the above two constraints cannot be perfectly met at anytime. If precise KNN is desired for each query, one should reveal his exact location and this violates the privacy constraint imposed on the problem (i.e., preserving user's location). Therefore an ideal approach should respect both constraints as much as possible i.e., it should be a very *close* approximation of S' (we will define the notion of closeness in Section 3.2) while keeping $P(Q)$, $P'(Q)$ and $\dot{P}(o_j)$'s as low as possible.

3.2 Space Encoding

In this section, we introduce our novel approach for protecting user's location from the malicious location servers. Our approach is based on transforming the static objects in the 2-D space as well as dynamic query points by mapping them to another space using a one-way transformation and addressing the query in the transformed space. As mentioned earlier we address the issue of location privacy in the context of location-based services and thus focus on the 2-D space of static objects (i.e., points of interest) and dynamic query points (i.e., users). Transforming such a 2-D space requires using a one-way function to map each point from the original space to a point in the transformed space. A transformation is one-way if it can be easily calculated in one direction (i.e., the forward direction) and is computationally impossible to calculate in the other (i.e., backward) direction [20]. The process of transforming the original space with such a one-way mapping can be viewed as *encrypting* the elements of the 2-D space. With this view, in order to make decryption possible the function has to allow fast computation of its inverse given some extra knowledge, termed *trapdoor* [19]. In practice, many one-way transformations may be reversible even without the knowledge of the trapdoor but the process must be too complex (equivalent to exhaustive try) to make such transformation computationally secure.

Therefore , as depicted in Figure 1, any one-way transformation which respects the proximity of the original space can replace the first black box in Figure 1 to make the location server *privacy aware*. We view such one-way transformations as modules that *encode* the original space into another space which is capable of addressing encoded transformed queries. In order to enable decoding of query results one should define the notion of a trapdoor or a *key* for the space encoding module. In this paper we use the

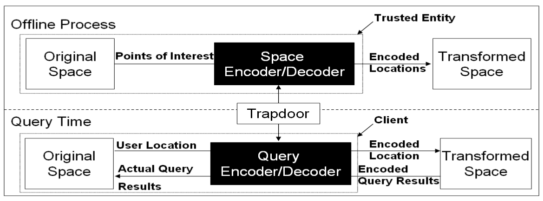

Fig. 1. Space Encoding.

properties of our mapping function as the trapdoor for fast decryption of query results. Such trapdoor will be only provided to the user to reverse the encoded results and get the response set back in its original format.

Note that hiding the location of the query point is different than hiding its identity and the focus of this paper is on hiding the locations of the query points and result sets. No matter what type of space encoder is used, a user will only have to report its encoded position to the location server in order to get the exact location of her result set back. In Section 6 we will discuss how this property separates user anonymization issues from protecting user's location.

Identifying the right *space encoders* is very challenging because there are several one-way transformations which could be applied to a 2-D space of objects (e.g., random perturbation of points), however, the majority of such transformations do not respect the notion of distance and proximity . The transformations that respect such properties are the only candidates resulting in satisfactory KNN query in an encoded space. We term such transformations *complete KNN-invariant* if performing the KNN query in the transformed space and decoding the result set back to the original space, generates a result set exactly equal to the result set obtained from performing the query in the original space. However, as we will discuss in Section 4, our proposed approach generates an approximation of the actual result for each KNN due to its nature of reducing the dimensionality of data. Therefore we define a weaker notion of *closeness* for a transformation and call it semi KNN-invariant (or for simplicity KNN-invariant) if it yields satisfactory values for the two metrics introduced in the following definition.

Definition 5. Suppose the actual result of a KNN query, issued by a user located at point Q is $R = (o_1, o_2, \dots, o_K)$, and it is approximated by a transformation *T* as $R' = (o'_1, o'_2, \dots, o'_K)$. *T* is KNN-invariant if it yields acceptable values for the following two metrics:

Metric 1: The *Resemblance*, denoted by α, defined as

$$\alpha = \frac{|R \cap R'|}{|R|} \tag{1}$$

where $|R|$ denotes the size of a set R. In fact α measures what percentage of the points in the actual query result set R are included in the approximated result set R'.

Metric 2: The *Displacement*, denoted by β, defined as

$$\beta = \frac{1}{K}\left(\sum_{i=1}^{K} ||Q - o'_i|| - \sum_{i=1}^{K} ||Q - o_i||\right) \tag{2}$$

where $||Q - o_i||$ is the Euclidean distance between the query point Q and o_i. Therefore β measures how *closely* R is approximated by R' on average. Obviously, since R is the ground truth, $\beta \geq 0$.

Although there is no fixed threshold for acceptable α and β values, depending on the application and the scenario, certain values may or may not be considered satisfactory. In Section 7 we will evaluate our approach against these two metrics and will show that it uses an effective KNN-invariant transformation.

In this paper, we study an important class of transformations called space filling curves as candidate space encoders for our framework. Such curves have interesting properties which have made them a popular tool in different domains such as querying multi-dimensional data and image compression [12, 16]. It is important, however, to note that we are not claiming that space filling curves are the best possible encoders. In Section 3.3 we show how such space filling curves can be treated as one-way functions if certain properties of those curves are kept secret from malicious attackers.

3.3 Space Filling Curves

Introduced in 1890 by an Italian mathematician G. Peano [17], space filling curves belong to a family of curves which pass through all points in space without crossing themselves. The important property of these curves is that they retain the *proximity* and *neighboring* aspects of the data. Consequently, points which lie close to one another in the original space mostly remain close to each other in the transformed space. One of the most popular members of this class is Hilbert curves [7] since several studies show the superior clustering and distance preserving properties of these curves [3, 9, 12, 15].

Similar to [15] we define ${H_d}^N$ for $N \geq 1$ and $d \geq 2$, as the N^{th} order Hilbert curve for a d-dimensional space. ${H_d}^N$ is therefore a linear ordering which maps an integer set $[0, 2^{Nd} - 1]$ into a d-dimensional integer space $[0, 2^N - 1]^d$ as follows:
$H = \ell(P)$ for $H \in [0, 2^{Nd} - 1]$, where P is the coordinate of each point in the d-dimensional space. We call the output of this function its *H-value* throughout the paper. Note that it is possible for two or more points to have the same H-value in a given curve.

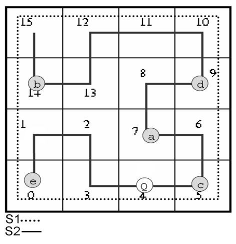

Fig. 2. A ${H_2}^2$ Pass of the 2-D Space.

As mentioned above, our motivating application is location privacy and therefore we are particularly interested in 2-D space and thus only deal with 2-D curves $(N = 2)$. Therefore $H = \ell(X, Y)$ where X and Y are the coordinates of each point in the 2-D space. Figure 2 illustrates a sample scenario showing how a Hilbert curve can be used to transform a 2-D space into H-values. In this example, points of interest (POI) are traversed by a second order Hilbert curve and are *indexed* based on the order they are visited by the curve (i.e., H in the above formula). Therefore, in our example the points a, b, c, d, e are represented by their H-values 7, 14, 5, 9 and 0, respectively. Depending on the desired resolution, more fine-grained curves can be recursively constructed as depicted in Figure 3.

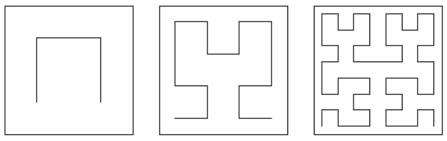

Fig. 3. First Three Orders of Hilbert Curves.

As we will show in Section 7, the most interesting feature of Hilbert curves is how they can act as KNN-invariant transformations with satisfactory values of α and β when used in location-based services. This property suits our approach as we are interested to address the KNN query in a transformed space and still get satisfactory results. Furthermore, another important property of a Hilbert curve that makes it a very suitable tool for our proposed scheme is that ℓ becomes a one-way function if the curve parameters are not known. These parameters, which collectively form a *key* for this one-way transformation, include the curve's starting point (X_0, Y_0), curve orientation θ, curve order N and curve scale factor Γ. We term this key, *Space Decryption Key* or SDK where $SDK = \{X_0, Y_0, \theta, N, \Gamma\}$.

Therefore a malicious entity, not knowing this key, has to exhaustively check for all combinations of curve parameters to find the right curve by comparing the H-values for all points of interest. As we show in Theorem 1, we make it computationally impossible to reverse the transformation and get back the original points. Even a nominal error in approximating curve parameters will generate a completely different set of H-values. We now prove two important properties of our approach which give more insight on the security of our proposed method.

THEOREM 1. The complexity of a brute-force attack to find the transformation key discussed above is $O(2^{4p})$ where p is the number of bits used to discretisize each parameter.

PROOF. In order to accurately find the curve's starting point, it should exactly lie on the intersection of two edges (lines) coming from each of the X and Y axes. Therefore one has to locate the exact values of both X_0 and Y_0 in the continuous domain of X and Y axes. Theoretically, the probability of finding the right value for the above two parameters in a continuous space is zero. However, in real world scenarios, the attacker can approximate X_0 and Y_0 by constructing the finest grid possible to guarantee that his best guess (X_0', Y_0') located at an intersection of two edges, lies very close to (X_0, Y_0) so that $|X_0 - X_0'| \leq \varepsilon$ and $|Y_0 - Y_0'| \leq \varepsilon$. When ε is sufficiently small then replacing (X_0, Y_0) with (X_0', Y_0') generates a set of H-values indifferentiable from the original set. The attacker should thus search the entire space exhaustively for a very close approximation of this starting point. Using p bits the attacker can generate 2^p candidate values on each axis. Therefore, assuming a square region covering all

POI's, the attacker's entire search space for the starting point will have $2^p * 2^p$ elements. Similarly, the entire continuous $360°$ space for θ should be discretized to the finest possible extent to ensure that $|\theta - \theta'| \leq \varepsilon$ for at least one value of θ'. With q bits, that attacker can generate 2^q different candidate values of θ each corresponding to a curve orientation. The curve scale factor Γ is a continuous number between 0 and 1 and thus similarly, r bits can divide the 0 to 1 range into 2^r values each can approximate Γ so that $|\Gamma - \Gamma'| \leq \varepsilon$ for at least one value of Γ'. Assuming N different possibilities for the curve order, the entire solution space will have $2^p * 2^p * 2^q * 2^r * N$ elements. Assuming $2^q = O(2^p)$ and $2^r = O(2^p)$ and since $N \ll 2^p$, the complexity of an exhaustive search is $O(2^{4p})$ where p is the number of bits used by the attacker to represent each parameter. ❑

Note that for a given N, there is an upper bound for p, after which there is no reason to increase p further because ε becomes sufficiently small to estimate X_0 and Y_0 accurately. However, by simply increasing N to $N + 1$ we can make the curve twice condense in each direction that results in a new threshold of $\varepsilon' = \frac{\varepsilon}{2}$ for the curve's starting point and similar tighter thresholds for other curve parameters. Therefore, a linear increase in N will make ε exponentially smaller and thus p should increase linearly with N as well to ensure close approximation of curve parameters. However, as Theorem 1 shows, increasing p will result in an exponential increase of the search space. Consequently, N is chosen large enough to make reversing an ${H_2}^N$ mapping impossible and thus to make H act as a one-way mapping. Hence, we consider this transformation as a *space encryption scheme* whose key is the curve parameters (i.e., *SDK*).

THEOREM 2. Using an ${H_2}^N$ Hilbert curve to encode the space satisfies the *a-anonymity, u-anonymity* and *result set anonymity* properties defined in Section 3.1.

PROOF. An ${H_2}^N$ fills a $2^N * 2^N$ grid in the 2-D space visiting each point exactly once. Theorem 1 states that having an H-value for the query point Q, one cannot reverse the process to find $\ell^{-1}(X_Q, Y_Q)$ because H is one-way for large values of N (i.e., curve degree) and thus Q cannot be located anywhere in the grid. With n and A being the total number of POI's and the entire region covering these n objects, respectively (see Definition 1), there are $2^p * 2^p$ equiprobable choices for the location of Q and thus $P'(Q) = 1/2^{2p} = \frac{1}{area(A)}$. Furthermore, since no information beyond the H-value of the query point is needed to resolve the query, Q could be issued by any user u_i and thus $P(Q) = \frac{1}{M}$ where M is the total number of users. Finally, for each static object o, $\ell^{-1}(X_o, Y_o)$ cannot be found and thus $\dot{P}(o_i) = P'(Q) = 1/2^{2p} << 1/n$ because $2^{2p} >> n$. ❑

4 2-Phase Query Processing

Making a query processing engine privacy-aware based on our idea of space transformation discussed above, requires a two-step process consisting of an offline encryption of original space followed by online query processing. First, during an offline process, necessary data structures and encryption schemes are utilized to encode the space of POI's. Next, during an online process, the query is resolved in the transformed space and is then decoded to obtain the original points satisfying the query in the 2-D space. The following sections describe the details of these two phases and the modules performed in each phase.

4.1 Offline Space Encryption

Figure 4 depicts Algorithm 1, the Offline Space Encryption algorithm. The first step of this phase is to choose the curve parameters from which the curve will be constructed and the value of *SDK* will be determined. These parameters are listed in Section 3.3. Next, assuming the entire area covering all points of interest is a square S_1, an $H_2{}^N$ Hilbert curve is constructed starting from (X_0, Y_0) in a (possibly larger) square S_2 surrounding S_1 until the entire S_2 is traversed (see Figure 2). After visiting each point P, its H-value = $\ell(P.X, P.Y)$ is computed using *SDK*. We use an efficient bitwise interleaving algorithm from [3] to compute the H-values for points of interest. This process is performed once for all points of interest and thus at the end of this step, a look-up table DB which consists of H-values for all POI's is constructed. Note that the size of DB is only dependant upon the number of POI's to be indexed and not the size of the region in which they are located. The result of applying Algorithm 1 on the example from Figure 2 looks like the following look-up table: $DB = \{(0), (5), (7), (9), (14)\}$ where each element is the point's index in the curve (i.e., its H-value).

```
KNN-CreateIndex (S) {
Generate SDK=F(X0,Y0,θ,Γ);
While (S ≠ Ø) {
   nextPoi=S.getNextPOI();
   H= ℓ(nextPoi.X,nextPoi.Y);//using SDK
   DB=DB U (H);
   }
}
```

Fig. 4. Offline Space Encryption.

4.2 Online Query Processing

Algorithm 2 (Figure 5) summarizes the online query resolution process and its two modules KNN-Encode and KNN-Resolve. Using the look-up table DB, we can now show how the result of a KNN query is evaluated in the transformed space. For each query point Q located at position (X_Q, Y_Q), KNN-Encode uses *SDK* to compute $H = \ell(X_Q, Y_Q)$. The value of H, along with K (i.e., the number of desired nearest neighbors), is all KNN-Resolve needs to resolve a query using DB. During this phase we begin searching from both directions in DB starting from $\ell(X_Q, Y_Q)$ until K closest matches are found. Note that these matches are nothing but K (distinct or overlapping) H-values. Knowing *SDK*, KNN-Encode transfers the result set back to the original 2-D space, using H^{-1} to decrypt the H-values of all points in result set. To illustrate, in our example, having $K = 3$, and $Q = (2, 0)$, KNN-Encode computes $H = 4 = \ell(2, 0)$ and calls KNN-Resolve$(4, 3)$ to obtain $R = \{(0), (5), (7)\}$. Next, H^{-1} is applied to all above H-values to obtain their original 2-D coordinates.

We can now derive the complexity of the KNN-Resolve module which represents the overall query processing complexity. As discussed in Section 4.1, an $H_2{}^N$ Hilbert curve divides the entire space into 2^{2N} equally spaced indices. This division, results in an average density of $\frac{n}{2^{2N}}$ POI's per each H-value where n is the total number of

```
KNN-Encode(K,X_Q,Y_Q) {
   H= ℓ(X_Q,Y_Q);//using SDK
   R= KNN-Resolve (H,K);
   While {R ≠ Ø}
      {
         O=R.GetNextH-Value();
         {Poi.X, Poi.Y}= ℓ^-1(O) //using SDK
         Result = Result U Poi;
      }
Result = Sort (Result);
}
```

```
KNN-Resolve (H,K) {
IndexMore=H; IndexLess=H-1;
Count=0; R= Ø;
While (Count<K)
 {
  if (DB.containsKey(qIndexMore))
   While (qIndexMore.HasMoreElem & count<K){
    R= R U qIndexMore.NextPOI; count++; }
  if (DB.containsKey(qIndexLess))
   While (qIndexLess.HasMoreElem & count<K){
     R= R U qIndexLess.NextPOI, count++; }
qIndexMore++; qIndexLess--;
}
Return R={ℓ(o_1), ℓ(o_2)..., ℓ(o_k)} ;
}
```

Fig. 5. Online KNN Query Resolution.

POI's. Therefore, finding the K closest objects in this space will on average require only $K \times \frac{2^{2N}}{n}$ H-value comparisons. Therefore, the overall complexity of our online query processing scheme is $O(K \times \frac{2^{2N}}{n})$ compared to $O(n)$ if traditional encryption schemes were used. Also the communication complexity of our scheme is $O(K)$ since the result set generated by KNN-Resolve includes only the K matching points compared to an $O(log(n))$ complexity using traditional encryption schemes and to K-anonymity or cloaking approaches in which the query result has to be generated for $K-1$ other points or an entire region, respectively. The efficiency of our query processing algorithm is more pronounced with real-world datasets where the value of n is significantly large.

Notice that the order of the result set might not be accurate because there are cases in which elements with smaller difference in H-values to Q are actually located further from it compared to other objects with larger H-value difference. However, this is essentially resolved by simply having the entire result set back in its original format. Knowing Q's location, KNN-Encode sorts the result set in the correct order. This is a very efficient process given the relatively small values of K. In addition, it is important to note that the result set of a KNN query might not precisely match the actual K nearest neighbors of a user because of loss of a dimension in the transformed space. Depending on different curve parameters and the data distribution, the accuracy of the result may vary. In Section 7, we conducted several experiments with real-world datasets and show that the *Resemblance* and *Displacement* values are acceptable for many real applications.

5 Dual Curve Query Resolution

Using a single Hilbert curve as a space encoder for KNN query processing discussed in Section 4 has two major drawbacks. We first discuss these two drawbacks and then introduce our *Dual Curve Query Resolution* approach or *DCQR* which overcomes the weaknesses of the former scheme and generates significantly more satisfactory results.

A closer study of Hilbert curves reveals two important properties of such curves. First, consider the 1st degree curve of Figure 6 (the left image). The curve naturally is constructed by traversing a U-shaped pattern. Regardless of its orientation, such a curve will fill the space at a specific direction at any given time sweeping the space in a clockwise fashion. Starting from the first degree curve of Figure 6, the curve misses one side in its first traversal. As the curve order grows, the number of missed sides grows exponentially as well so that an ${H_2}^N$ curve misses $M = 2^{2N} - 2^{N+1} + 1$ sides of a $(2^N - 1)$ by $(2^N - 1)$ grid. The above property of the curve will make H-values of certain points farther as N increases. For instance the Euclidean distance between points a and d is similar to the that of points b and c in the original 2-D space, however due to the above property, a and d's H-values will be significantly further from each other as compared to H-values of b and c. This difference grows exponentially as N grows. Therefore points closer to two quadrants of the space (i.e., the first and last quadrants filled by the curve) will be *spatially furthest* from one another in the transformed space.

The second drawback of using a single Hilbert curve is due to the fact that such space-filling curves essentially reduce the dimensionality of the space from 2 (or in general case N) to 1. Naturally, each element in the 1-D space constructed by the Hilbert curve will have two nearest neighbors compared to the original case where each element (except those at the edges) has four (or in general case $2N$) nearest neighbors. Therefore as [10] suggests, in the best case scenario, only half of these nearest neighbors in 2-D space will remain a nearest neighbor of the same point in the transformed 1-D space.

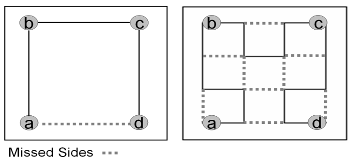

Fig. 6. Missed Sides of 2 by 2 and 3 by 3 Grids for ${H_2}^1$ and ${H_2}^2$, respectively.

The above two properties result in a loss of precision and thus a negative effect on overall quality of returned results. We mitigated this issue by replicating the same curve and rotating it 90 degrees. Our intuition is to index the same data simultaneously by two perpendicular curves and ask each one independently to resolve a KNN query using modules discussed in Section 4. Having two different result sets in the original domain, we merge the results and choose the K best candidates among the $2K$ points of the sets.

We now discuss how *DCQR* ensures a better quality of results. By rotating the degree N curve, all lower degree curves constructing the main curve will be rotated as well. At each curve order, the curve rotation ensures that the missed sides generated by the discontinuation of the curve (such as the missed sides between points a and d in Figure 6), will be covered by the rotated curve. Therefore, the points deemed spatially far from each other in one curve will be indexed correctly in the other curve. This will address the first issue when using Hilbert curves for indexing. *DCQR* also mitigates the effect of the second property discussed above by transforming the 2-D space to two 1-D spaces. Therefore each point will now have two nearest neighbors in each curve. It is important however, to note that these two neighbor pairs can (and do) often have overlaps and that is the main reason the dual curve approach will generate (significantly more accurate) approximate answers. Furthermore, with regards to complexity, knowing the first curve's *SDK* makes it easy to derive the key for the second curve (curve order and scale factor are the same while curve orientation and starting point are rotated 90 degrees). Therefore, the complexity of finding *DCQR*'s keys differs from what we derived for a single curve approach in Section 3.3 by a constant factor. The query processing discussed in Section 4 should also be modified slightly to work with the new dual curve scheme as follows (note that these modifications do not change the query computation and communication complexities derived in Section 4.2).

5.1 Offline Space Encryption for *DCQR*

During this phase, we again assume that the entire static objects set is located inside a square $S1$. Consequently two Hilbert curves ${H_2}^N$ and ${H_2'}^N$ are constructed based on *SDK* to sweep the (possibly larger) square $S2$ (surrounding $S1$), until the entire $S2$ is traversed. Visiting each point, H and H' will compute $\ell(X,Y)$ and $\ell'(X,Y)$ respectively in the similar fashion discussed in Section 4.1. After this process is performed once for all POI's, the two sequences of H and H$'$-values will form two separate look-up tables DB and DB'.

5.2 Online Query Processing for *DCQR*

Similarly, the query processing follows the logic from Section 4.2 with the difference that for each query point Q, we compute $H = \ell(X_Q, Y_Q)$ and $H' = \ell'(X_Q, Y_Q)$ using *SDK* and *SDK*$'$, respectively. We then initiate two parallel query resolution schemes applying H-value and H$'$-value to DB and DB$'$, respectively and simultaneously retrieve K closest matches for each curve separately. Similar to Section 4.2, we decrypt the two results sets and choose the K best candidates (based on their Euclidean distance to Q).

6 Proposed End-to-End Architecture

In previous sections, we showed in detail how we can utilize Hilbert curves as space encoders to blindly resolve KNN queries. As we mentioned earlier, the focus of this paper is on hiding locations and not *identities* of static objects or query points. However, in order to propose a complete solution, we briefly discuss how we can extend our proposed scheme to deal with non-spatial attributes of each POI (such as its identity or name) in the following way; Similar to *SDK*, we define a *Textual Decryption Key* or *TDK*, which is used to encrypt (decrypt) the non-spatial attributes of each POI during the offline space encryption (online query resolution) phase. Therefore, during the offline phase, in addition to the steps discussed in Section 5.1, we generate *TDK* and *TDK*$'$ and use them to encrypt the textual attributes of each point P represented by $E(P.T)$ where E is the function used to encrypt $P.T$ (the textual attributes of P). The above modifications will add a new attribute to DB and DB' to include any textual information of POI's. Therefore DB and DB' become look-up tables with the schemas (H-value, E(o.T)) and (H$'$-value, E$'$(o.T)), respectively.

6.1 End-to-End Query Processing

We are now ready to explain how a KNN-invariant one-way transformation can be used for blinding KNN queries in location-based services. The *client* (e.g., a portable device)

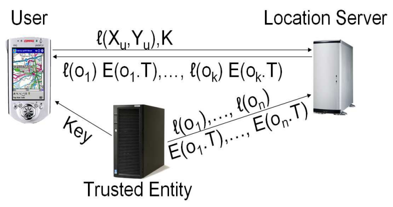

Fig. 7. *DCQR* Architecture for KNN Query Processing.

issues a K-nearest-neighbor (KNN) query and provides its own location. Without loss of generality, we assume the client location is a point and is identified by two values such as its latitude and longitude. In order to make the location server privacy-aware, we first assume the architecture of Figure 7 which details the sequence of client-server communications required in order to resolve a KNN query and use the algorithms discussed in Section 5 to modify the classic location-based services architecture in the following three ways:

1) A *trusted entity* is added to the architecture. The main task of the trusted entity is to perform the KNN-CreateIndex module once and to create and update their encoded indexes and identities. A second functionality of the trusted entity is to provide users with (*SDK*,*SDK′*) and (*TDK*,*TDK′*) pairs required to decrypt query results. We refer to these four values as *Key Pairs*. Finally, the trusted entity provides the location server with the two look-up tables DB and DB' instead of the original dataset and keeps the two key pairs secret from the location server. Note that unlike an anonymizer, the trusted entity is not involved in the query processing.

2) Users will perform the KNN-Encode module and use the two key pairs embedded in their devices to decrypt the result set returned to them from the location server and get back the location of the returned points as well as their textual attributes. Note that in order to prevent users from being able to access the encrypted result set received from the location server and learning the transformation, the key pairs should be embedded in tamper-proof devices. Furthermore, in order to remain anonymous, users generate a random *session-id* for each KNN query request in order to enable the client and server to communicate with each other during the course of each KNN query.

3) The un-trusted location server will perform the KNN-resolve module to construct the two results sets and returns them to the user.

7 Experimental Evaluation

We have conducted several experiments to evaluate the performance of our proposed approach. The effectiveness of *DCQR* is determined in terms of 1) the effect of the curve order N on our proposed indexing, 2) accuracy of the result sets in terms of the Displacement and Resemblance metrics defined in Section 3.2, using *DCQR* instead of a single curve, and 3) *DCQR*'s vulnerability to attacks. We were unable to compare our approach with other approaches discussed in Section 2, because they mostly evaluate performance, based on the size of the K-anonymity set, the size of the cloaked region or the effectiveness of the anonymization techniques used and our approach is free of these metrics and satisfies stronger privacy metrics defined in Section 3. We have also performed other experiments that investigate the effect of other key parameters on quality of indexing and demonstrate our fast overall system response time (typically less than 0.5 seconds even for large values of K and N). We do not discuss these experiments here due to lack of space and we plan to fully investigate them in an extended version of this paper. Our experiments are performed on a real-world dataset obtained from NAVTEQ covering a 26 mile by 26 mile area surrounding the city of Los Angeles which contains more than 10000 restaurants. Experiments were run on an Intel $P43.20$ GHz with 2 GB of RAM.

7.1 The Curve Order N

In our first set of experiments, we evaluate the effectiveness of our proposed indexing technique. It is important to analyze the curve behavior for different values of N (i.e., curve order) and to decide on the value of *SDK* and use it throughout the rest of our experiments. For the first set of experiments, we measure the effectiveness of two ${H_2}^N$ curves in indexing POI's for fixed values of $X_0 = Y_0 = \theta = 0$ and $\Gamma = 1$ and varying N from 1 to 15 for the first curve (note that *SDK* of the dual curve, i.e., *SDK′*, can be derived from *SDK*). We measure the minimum and average number of POI's which are assigned the same H-value for each value of N. It is clear that having a large number of POI's with the same H-value has a negative effect on Resemblance and Displacement metrics because the location server has no way of choosing a *closer* point, in a set of POI's with the same H-value while responding to a KNN query. Varying N, makes an entirely different curve and thus changes the assignments of H-values to POI's significantly. Figure 8 shows how POI/H-value changes with N. It suggests acceptable values of this number (i.e., POI/H-value$\leq$ 2) for curves with $N \geq 8$. Our next experiments confirm this intuition.

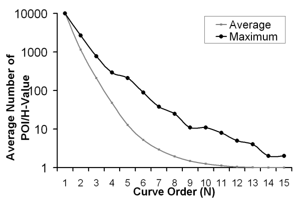

Fig. 8. Curve Order Vs. H-Values

7.2 The Single Curve Approach vs. *DCQR*

In the second sets of experiments we first compare the single curve approach with *DCQR* in terms of the Displacement and Resemblance metrics defined in Section 3.2. As Figure 9 illustrates, for different curve orders (i.e., N) and different values of K (i.e., different KNN queries), *DCQR* outperforms the single curve approach for both metrics, achieving lower average Displacement and higher Resemblance values.

Next, we evaluate *DCQR* using the same metrics. As shown in Figure 10, for a fixed value of $N = 12$, an increase in K improves the Resemblance while it does not have a significant effect on the Displacement. The reason is that as K increases, the result set size grows twice as fast (since using *DCQR*, its size is 2K for each KNN query) which in

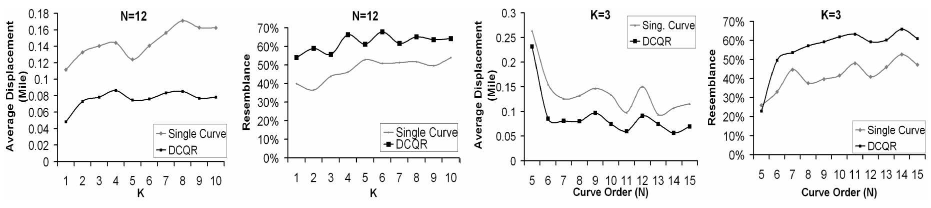

Fig. 9. Comparing Single Curve Approach vs. *DCQR* for Different Values of K and N

turn increases the chance of visiting the right points as we move on the curve. However, searching for more POI's on the curve also causes moving further away from the query point's index on the Hilbert curves which might increase the probability of hitting a missed side and thus including a false positive in the result set. However this negative effect is nominal and the Displacement stays less than 0.08 mile for all possible values of K. Similarly, for a fixed value of K=3, while the Displacement takes satisfactory values (less than 0.09 mile on average) for $N \geq 8$, Resemblance usually improves as N grows, confirming our intuition from the first set of experiments. Similar trends were observed for other fixed values of K and N.

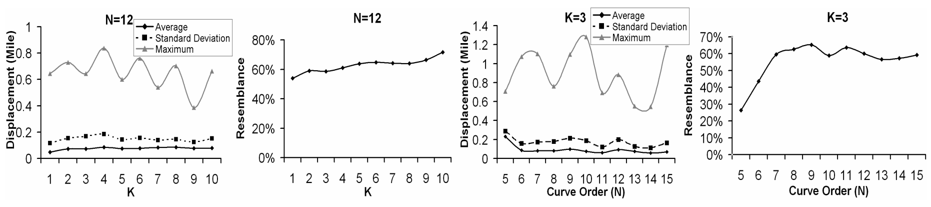

Fig. 10. DCQR Performance vs. K and N

7.3 *DCQR*'s Vulnerability to Attacks

Our last set of experiments empirically evaluates the vulnerability of our proposed approach against malicious attackers to confirm the hypotheses discussed in Section 3.3 for the one-wayness of transformations used in *DCQR* and the security of *SDK* based on the following two extreme scenarios. First we assume the malicious location server (which is capable of becoming the most powerful attacker due to its access to DB and DB'), has somehow gained precise knowledge for the values of X_0, θ, Γ and N and only needs to find Y_0. Using p bits, it divides the Y-axis to 2^p distinct values hoping to get close enough to Y_0. For each of its guesses Y_0', the location server forms an *SDK* and performs the KNN-CreateIndex module to compare the resulting look-up table against DB (or DB') and measures the Resemblance metric to evaluate Y_0'. Figure 11 (left) illustrates the result of this attack for p taking 12, 15, 18 and 22 bits (which correspond to a minimum of $10^{-2}, 10^{-3}, 10^{-4}$ and 10^{-5} mile displacement between Y_0 and Y_0'), respectively. The location server's best guess is where it uses the maximum number of bits (i.e., $p = 17$ and $|Y_0 - Y_0'| \simeq 1meter$) which results in a look-up table less than 10% similar to DB. Note that the location server does not even know which H-values belong to the above 10% subset of DB and thus even by getting very close to real curve parameters, the key cannot be compromised.

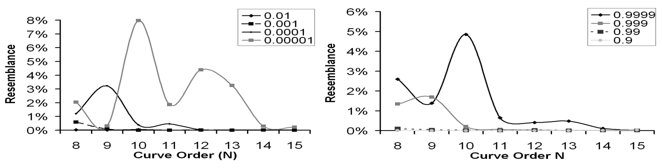

Fig. 11. Attacking SDK by Approximating $|Y_0 - Y_0'|$ (left) and $\frac{\Gamma}{\Gamma'}$ (right)

Similar to the above case, we now assume that the malicious location server knows the exact values of X_0, Y_0, θ and N and should only approximate the value of Γ with Γ'. Taking the same approach, the location server uses 4, 7, 10 and 14 bits so that the value of $\frac{\Gamma}{\Gamma'}$ approaches 0.9, 0.99, 0.999 and 0.9999, respectively. Figure 11 (right) shows that in the best case where it uses the maximum number of bits, (i.e., $p = 14$) the generated look-up table bears less than 5% similarity to DB again without the location server knowing the subset of points indexed accurately. Therefore the last two sets of experiments demonstrate the strong robustness of our proposed scheme against malicious attacks.

8 Conclusion and Future Work

In this paper, we discussed the problem of location privacy in location-based services. We studied the challenges of achieving location privacy and introduced a novel way of blindly evaluating KNN queries, an important class of spatial queries in location-based services, by using one-way space transformations to map objects and query points into an unknown space and evaluate the query in that space. The major contributions of our work can be summarized as follows:

- We proposed blind evaluation of queries using Hilbert curves as space encoders and introduced *DCQR*, our proposed Dual Curve Query Resolution approach and designed an $O(K \times \frac{2^{2N}}{n})$ computation and $O(K)$ communication algorithm which enables *DCQR* to resolve KNN queries in the transformed space (where n is the total number of POI's and N, the curve order, is a small constant.
- We introduced two new privacy metrics, *u-anonymity and a-anonymity*, which are much stronger and more generalized than the privacy constraints of commonly used K-anonymity and spatial cloaking based approaches.
- We analytically proved the one-wayness property of our space encoding technique and showed how *DCQR* achieved the result set anonymity as well as u-anonymity and a-anonymity metrics to become privacy-aware.
- We studied a set of powerful attacks based on the number of bits used to encode the space and empirically evaluated the resilience of *DCQR* against these attacks.
- We conducted extensive experiments to show the superior properties of our blind KNN query resolution scheme.

We intend to study other space mappings and identify new KNN-invariant transformations and propose efficient ways of turning them into space encoders allowing exact answers to be generated for KNN queries while still respecting the location privacy constraints discussed in this paper.

References

1. A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
2. C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In W. Jonker and M. Petkovic, editors, *Secure Data Management*, volume 3674 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2005.
3. C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 247–252, New York, NY, USA, 1989. ACM Press.
4. B. Gedik and L. Liu. A customizable k-anonymity model for protecting location privacy.
5. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*. USENIX, 2003.
6. M. Gruteser and X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy*, 2(2):28–34, 2004.
7. D. Hilbert. Uber die stetige abbildung einer linie auf ein flachenstuck. In *Math. Ann. 38*, pages 459–460, 1891.
8. P. Indyk and D. P. Woodruff. Polylogarithmic private approximations and efficient matching. In *Theory of Cryptography, Third Theory of Cryptography Conference*, pages 245–264, New York, NY, USA, 2006.
9. H. V. Jagadish. Linear clustering of objects with multiple atributes. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 332–342, Atlantic City, NJ, 1990. ACM Press.
10. H. V. Jagadish. Analysis of the hilbert curve for representing two-dimensional space. *Inf. Process. Lett.*, 62(1):17–22, 1997.
11. P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving anonymity in location based services. *A Technical Report TRB6/06, National University of Singapore*, 2006.
12. J. K. Lawder and P. J. H. King. Querying multi-dimensional data indexed using the hilbert space-filling curve. *SIGMOD Record*, 30(1):19–24, 2001.
13. M. F. Mokbel. Towards privacy-aware location-based database servers. In R. S. Barga and X. Zhou, editors, *ICDE Workshops*, page 93. IEEE Computer Society, 2006.
14. M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 763–774, Seoul, Korea, 2006. ACM.
15. B. Moon, H. v. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, 2001.
16. F. Pinciroli, C. Combi, G. Pozzi, M. Negretto, L.Portoni, and G. Invernizzi. A peano hilbert derived algorithm for compression of angiocardiographic images. In *Computers in Cardiology*, pages 81–84. IEEE Computer Society Press, 1991.
17. H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.
18. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.
19. M. R. Schroeder. *Number Theory in Science and Communication*. Springer-Verlag, 1984.
20. D. R. Stinson. *Cryptography, Theory and Practice*. CHAPMAN & HALL/CRC, 2002.